
DECAL TOKENWISE COMPRESSION

Sameer Panwar

Google DeepMind

sameerpanwar@google.com

ABSTRACT

This paper introduces DeCAL, a new method for tokenwise compression. DeCAL uses an encoder-decoder language model pretrained with denoising to learn to produce high-quality, general-purpose compressed representations from the encoder. DeCAL applies small modifications to the encoder, with the emphasis on maximizing compression quality, even at the expense of compute. We show that DeCAL at 2x compression can match uncompressed on several downstream tasks, with usually only a minor dropoff in metrics up to 8x compression, among question-answering, summarization, and multi-vector retrieval tasks. DeCAL offers significant savings where pre-computed dense representations can be utilized, and we believe the approach can be further developed to be more broadly applicable.

1 INTRODUCTION

Transformers (Vaswani et al., 2017) have proven very effective at a broad range of natural language tasks, but inference costs increase dramatically as sequence lengths increase. Efforts to address the cost often focus on speeding up the attention block (Tay et al., 2022), the main bottleneck, usually by sparsifying compute, but retaining all tokens since it is unknown which subsequent attention computations might attend to them. Pruning input tokens (Tang et al., 2024) is another strategy, but that leads to forgetting, which may sacrifice performance on certain tasks.

Alternatively, we can compress the input sequence by merging multiple token representations together, potentially retaining more information. Intuitively, token sequences should be fairly compressible given they are built from subwords and other predictable patterns. We know tokens contain nonuniform amounts of information; punctuation, for example, has less content than a long sentence-piece (Kudo & Richardson, 2018). Several subwords individually do not contain much information, but in sequence together they do; however, that information does not require their original space. Hence, the opportunity for compression is clear.

Approaches for tokenwise compression rely on variants of attention pooling (as in Funnel Transformer (Dai et al., 2020)) and convolution (as in CANINE (Clark et al., 2022)), primarily to deliver substantial training and inference speedups. The speedups do come at a cost to model performance, as we have observed in practice with the Funnel Transformer.

More recent work such as AutoCompressor (Chevalier et al., 2023), COCOM (Rau et al., 2025), and ICAE (Ge et al., 2024) share the goal of LLM context compression. They all derive compressive encoders from decoder-only LLMs in order to fit more information within the context budget of the LLM, as well as save inference time when the context chunks can be precomputed.

This work focuses on maximizing compressed representation quality, without making tradeoffs for immediate inference speedup, leaving compute optimization for later. We utilize the T5 framework, which natively supports encoder-decoder models, in order to readily explore the design space including compressive pretraining from scratch. This, however, makes it difficult to make apples-to-apples comparisons against decoder-only-derived LLM approaches. Instead, we build a proxy from the common elements of the recent LLM context compressors and compare against it, trained and evaluated identically in our environment.

We present DeCAL, a novel method for tokenwise compression. DeCAL employs an encoder structure in which we prepend an input-derived latent sequence to the encoder input. This shorter latent

sequence successively attends to the evolving input sequence, layer by layer, but only the latent is output. This structure forms the basis of the name DeCAL, short for Deep Cross-Attended Latents.

Our main contributions are as follows:

- We train DeCAL (as an encoder-decoder language model) from scratch with compression on a denoising language modeling task. We show that this is superior to the more typical approach of autoencoding (mixed with text continuation) after standard pretraining.
- We initialize DeCAL’s latent sequence with pooled input tokens combined with a learnable embedding vector, and show that this outperforms using the learnable vector alone.
- A simple alternative to DeCAL’s prepending m latent tokens to the input sequence is to instead perform attention pooling on the output layer down to m tokens. Though the latter is faster, we establish that it delivers inferior compression.
- We fine-tune DeCAL on context-intensive tasks, namely summarization and question answering, and compare different levels of compression, with only minor metrics dropoff at 8x compression. We also construct a proxy for typical LLM context compression approaches to compare against and determine that the proxy falls short of DeCAL performance.
- We illustrate the versatility of DeCAL by additionally applying it to multi-vector retrieval tasks.

2 RELATED WORK

There are a number of studies that involve compressing token sequences, though most do this internally for efficiency, not as an explicit usable output sequence.

Compressive Encoder-only. The Funnel Transformer (Dai et al., 2020) applies multiple steps of attention-pooling compression, followed by a single step of upsampling back to full sequence length (plus residual from the last full length layer). The upsampled layers are used for masked language pretraining, then discarded. In contrast, DeCAL does not compress the input stepwise in-line to the output, but rather cumulatively, alongside the input (as detailed under Methods). DeCAL uses a full-size decoupled autoregressive decoder to interpret the encoder’s compressed output and so more naturally handles more complex language modeling such as variable-length corrupted token spans.

Perceiver (Jaegle et al., 2021) uses a short latent sequence that repeatedly cross-attends to the original input sequence, thus requiring much less compute. DeCAL instead combines the latent with the original input and both sequences evolve layer by layer, requiring net additional compute (trading off for higher compression quality).

Internal compression. Several studies use compression internally, but not in any externally usable form. Hourglass (Nawrot et al., 2021) is a decoder-only architecture that resembles Funnel Transformer, with successive compression steps, but these are later followed by successive upsampling steps each with residual connections. This is done for the purpose of efficiency for long sequences.

CANINE (Clark et al., 2022) applies convolutions for both downsampling and upsampling (but no residual connections), with the goal of handling character-level inputs. Byte Latent Transformer (Pagnoni et al., 2024) groups input bytes into variable length patches which undergo attention pooling into one global token per patch before being upsampled with reversed attention pooling (the last full-length layer output serves as the attention queries). Both approaches apply the bulk of their compute in their compressed inner layers.

LongT5’s (Guo et al., 2022) sparse attention compresses each layer’s input using average pooling to serve as ephemeral KV sources for long-range attention.

History compression. Additional studies compress past history of decoders to cope with very long contexts. The Compressive Transformer (Rae et al., 2019) propagates past token history through recurrent encoding of token blocks, and then extends its memory capacity by compressing (via convolution) all but the last block. The Recurrent Memory Transformer (Bulatov et al., 2024) also performs recurrent encoding of token blocks, but carries over only a small fixed number of tokens per block, for very high compression to achieve 1M+ token contexts, but is very task-specific.

Pruning. A less common technique to shorten sequences is pruning tokens, such as PoWER-BERT (Goyal et al., 2020), which drops a few tokens every layer, knowing that at the output, only the CLS token is used. However, this is only added after pretrain and fine-tune, then an auxiliary model is used to identify how to drop tokens, resulting in a task-specific inference speedup. NUGGET (Qin & Van Durme, 2023) has an encoder-decoder format, but adds small FFNs on the encoder outputs to perform top-K selection. NUGGET is trained as an autoencoder, but is limited to 128 token blocks.

Soft prompts. Soft prompts (Li & Liang, 2021) involve adding a learnable prefix to a frozen model’s input, using a different prefix per task as an alternative to separate fine-tuning per task. This looks superficially like DeCAL in its learnable prefix, but is functionally orthogonal; DeCAL’s encoder outputs only the latent sequence, which is a function of the input sequence, unlike soft prompts.

Context compression. AutoCompressors (Chevalier et al., 2023) build on the Recurrent Memory Transformer (Bulatov et al., 2024) by keeping prior blocks’ summary vectors and successively concatenating them to finally obtain an effective compressed form that is used as a soft prompt. The In-context Autoencoder (ICAE) (Ge et al., 2024) works similarly, but compresses blocks in parallel instead of recursively, so blocks do not see their neighboring context. ICAE’s compression training consists of autoencoding with text continuation, in contrast to Autocompressor’s next token prediction. COCOM (Rau et al., 2025) develops further by unfreezing the decoder, and achieves a better compression ratio than ICAE, but encodes smaller chunks.

All of these works share the common approach of appending special context tokens to accumulate the compressed output representation, which is similar to DeCAL, though DeCAL initializes the latent tokens differently. All adapt from decoder-only LLMs, and do not apply denoising during their compression training, unlike DeCAL.

It is worth observing the commonality that the underlying compression operation in all of the above techniques is most often cross-attention from a small sequence to a larger one and, in some cases, convolution.

3 METHODS

The essential elements of DeCAL are:

- An encoder-decoder model configuration
- A compressive encoder structure with latent sequence prepended to the input
- Pooling of input tokens to seed the initial latent sequence state
- A denoising language modeling pretrain task

In an encoder-decoder language model, as in T5 (Raffel et al., 2020), the encoder provides the input context, while the decoder makes next-token predictions. This provides two very useful properties: 1) Unlike encoder-only masked-LM (Devlin et al., 2019), there are no indicators of how many tokens a compressed input token originated from, nor any residuals of the uncompressed input; the decoder is cleanly decoupled and must discern the contents of each compressed token. This puts more weight on the encoder to do a better job of compression and on the decoder to validate that. 2) Training examples have separate sequences for the encoder and decoder, allowing us to tune the amount of work each must do.

We find that the standard span corruption language modeling task in T5 (Raffel et al., 2020) works very well. For the encoder input, it removes spans of tokens, replacing them with sentinel tokens. The decoder target sequence is the complement, with just the tokens that had been removed, and sentinel tokens between them representing the token spans provided by the encoder. So despite the input to the decoder being compressed, it must still identify the sentinel tokens and the preceding and succeeding tokens, then denoise to output the correct span.

DeCAL is constructed as follows, using a standard T5 encoder-decoder:

- We take $\mathbf{x} = [x_1, \dots, x_n]$ as the input embeddings to be compressed.
- To create the initial latent sequence $\mathbf{l} = [l_1, \dots, l_m]$ where $m < n$ (giving us a compression ratio $C = n/m$),

- We start with a learnable vector v repeated m times.
- We then add the input embeddings, pooled from length n to m by pooling function P . In this work, for P , we use a strided mean pool with window size C (same as Funnel Transformer’s (Dai et al., 2020) attention query in its attention pooling).
- Lastly, we apply a layer norm.
- Thus $\mathbf{l} = \text{LayerNorm}(v_{\times m} + P(\mathbf{x}))$
- The input to the encoder is $\mathbf{l} \oplus \mathbf{x}$, thus longer than the original input, as shown in Figure 1.
- The encoder is internally unmodified, so there are negligible additional parameters. Note that both the \mathbf{l} and \mathbf{x} subsequences are able to attend to each other as their representations are updated in each layer.
- We assign the position index of tokens in \mathbf{l} to be the integer mean of the position indices of the corresponding tokens in \mathbf{x} . This is used for T5 relative position calculations.
- The encoder output is exclusively the m output embeddings corresponding to \mathbf{l} , as shown in Figure 1.
- The decoder is unchanged.

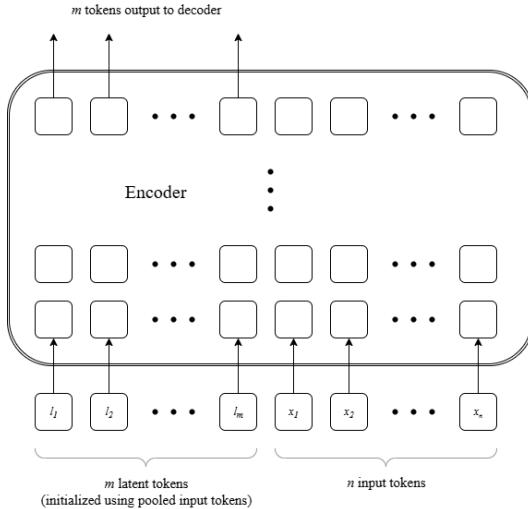


Figure 1: Diagram of DeCAL encoder input and output for compressing n input tokens to m output tokens.

In this scheme, we are logically assigning each token in \mathbf{l} to a span of tokens in \mathbf{x} , forming the correspondence of which \mathbf{x} tokens we expect to be compressed into which latent token. We initialize \mathbf{l} on this basis, but note we do nothing further with this correspondence, such as attention masking, etc. In our experiments, we only use a constant C , but the design can easily accommodate variable length mappings, such as used in BLT (Pagnoni et al., 2024). \mathbf{l} starts only from initial token embeddings, but as the \mathbf{x} subsequence is updated after each transformer layer, the \mathbf{l} subsequence can attend to \mathbf{x} as well as itself, building up a compressed contextualized representation. Likewise \mathbf{x} is able to attend to \mathbf{l} , though we haven’t tested its utility. We observe that this approach increases compute cost due to the increase in encoder input length; interestingly, this means compressing more (shorter \mathbf{l}) is lower compute than compressing less (longer \mathbf{l}), which has to be taken into account when trading off compute against compression quality.

The encoder maintains constant compression C , scaling m as necessary if n changes; e.g. if after pretraining at 4x and 1024 token inputs for example, we then fine-tune the model on a task with 8192 token inputs, the encoder will output 2048 tokens.

It would seem logical that performing compression alongside a normal transformer stack would require differing parameters; we tried having separate parameters for the transformer blocks handling the latent tokens, however, this did not produce better results, so further tweaking may be required to continue down that path.

4 EXPERIMENTS

4.1 SETUP

All our experiments used the T5.1.1¹ Large encoder-decoder, pretrained from scratch on 128M examples from C4 with a sequence length of 1024, and the default span corruption configuration of 15% corruption, which means 85% of the training tokens are fed to the encoder and 15% to the decoder. We used 10k steps of linear warmup and used a 4x larger relative position `num_buckets` and `max_distance` than T5’s default, since that worked best. The DeCAL models were trained identically, with the model structure as described under Methods and compression ratios varying from 2x to 8x.

4.2 ENCODER-DECODER EXPERIMENTS

4.2.1 DATASETS

Since this work is focused on compressing encoder inputs, we focused on summarization and question-answering tasks, which are naturally context-intensive, to evaluate compression effectiveness. For summarization, we used the arXiv (Cohan et al., 2018) and PubMed (Cohan et al., 2018) datasets, for which the document is the encoder input and the abstract is the target summary. We also included CNN / DailyMail (Nallapati et al., 2016), where the news article is the encoder input and the article’s summary bullets are the target summary, as well as MediaSum (Zhu et al., 2021), where interview transcripts are the encoder input and their topic and overviews served as the target summary.

For question-answering, we used HotpotQA (Yang et al., 2018) and TriviaQA (Joshi et al., 2017), for which the question and article context are given to the encoder, and the (first) answer is the target. We fine-tuned on 6M examples, except arXiv, for which we trained on 12M examples, since that task took longer to converge. We limited context to 4096 tokens for all tasks.

4.2.2 LANGUAGE MODELING

We tested DeCAL with compression ratios of 2x, 4x, and 8x. Since these ratios get quite aggressive, it can be instructive to see how these impact encoder-decoder pretraining. As shown in Table 1, the DeCAL target token prediction accuracy at 2x is able to match the (uncompressed) T5 Large baseline, while 4x is sufficiently close that further training may close much of the gap. At 8x, accuracy takes another comparable step down. Training at very high compression ratios of 16x and higher proved unstable during pretraining.

	Accuracy
Baseline	70.2
DeCAL 2x	70.4
DeCAL 4x	69.2
DeCAL 8x	68.0
DeCAL 16x	unstable

Table 1: Span corruption pretraining accuracy for different levels of DeCAL compression against no compression.

4.2.3 SUMMARIZATION

Table 2 shows ROUGE metrics for the summarization datasets. As with pretraining, we see that DeCAL 2x is always comparable to the baseline, while 4x is only modestly behind. 8x is a notable step lower, but not significantly considering the decoder only cross-attends to 512 tokens versus

¹https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md#t511

4096 uncompressed. We speculate DeCAL compression does favorably here because enough of the general semantics of the full input is preserved for the purpose of abstractive summarization.

	arXiv			PubMed		
	R-1	R-2	R-L	R-1	R-2	R-L
Baseline	45.6	18.9	41.6	48.5	23.1	45.1
DeCAL 2x	45.3	18.9	41.4	48.6	23.1	45.2
DeCAL 4x	44.9	18.6	41.0	47.9	22.5	44.5
DeCAL 8x	44.0	17.9	40.1	47.2	21.4	43.8
	CNN / DailyMail			MediaSum		
	R-1	R-2	R-L	R-1	R-2	R-L
Baseline	42.9	20.7	37.3	35.1	18.5	32.1
DeCAL 2x	43.2	20.9	37.5	35.0	18.3	32.0
DeCAL 4x	42.2	20.1	36.7	34.1	17.6	31.2
DeCAL 8x	41.4	19.1	35.9	32.2	16.2	29.8

Table 2: ROUGE metrics on summarization tasks for different levels of DeCAL compression against no compression.

4.2.4 QUESTION-ANSWERING

For the question-answering datasets, presented in Table 3, we see again that DeCAL 2x can fully match the baseline. DeCAL 4x and 8x gradually drop off, but much more so for HotpotQA, likely due to its multi-hop reasoning characteristic. This range of compression ratios gives us a way to trade off compression versus model performance, all the way to 8x.

	TriviaQA		HotpotQA	
	EM	F1	EM	F1
Baseline	76.8	79.1	65.1	79.2
DeCAL 2x	77.0	79.2	65.2	79.5
DeCAL 4x	76.0	78.3	62.9	77.0
DeCAL 8x	75.1	77.2	60.3	74.4

Table 3: SQuAD metrics on question-answering tasks for different levels of DeCAL compression against no compression.

4.2.5 COMPARISONS

Having a basis of comparison with prior work is useful in placing new work in context. The works that are most similar to DeCAL are LLM context compressors. However, these are built on a myriad of frameworks, have varying model sizes, maximum chunk lengths, and evaluation tasks. We feel it is more beneficial to instead extract the common elements of those designs to construct a proxy of these current approaches and compare against it. This eliminates extraneous effects from arbitrary differences, such as training data, token limits, LLM input packing strategies, etc.

Using AutoCompressor (Chevalier et al., 2023), COCOM (Rau et al., 2025), and ICAE (Ge et al., 2024) as reference, the common key elements are as follows:

- These all use learnable special tokens appended to the input to aggregate the compressed information.
- COCOM and ICAE both use autoencoding mixed 1:1 with text continuation for compression training. We use C4 for these, for 26M examples, and start from our baseline (uncompressed) model.

-
- These all derive from a decoder-only model, and so are subject to causal attention masks.

We will call the proxy built as above *CCProxy*.

Separately, an obvious alternative to prepending a latent sequence to the input is attention pooling the encoder’s final output. This has lower compute, so it is also worth comparing. We refer to this alternative as *AttnPool* in our results, and only test 2x compression. We train it identically to the DeCAL experiments, and we use mean pool (stride 2) to initialize the query of the attention pool layer.

	MediaSum			HotpotQA	
	R-1	R-2	R-L	EM	F1
Baseline	35.1	18.5	32.1	65.1	79.2
DeCAL 2x	35.0	18.3	32.0	65.2	79.5
DeCAL 4x	34.1	17.6	31.2	62.9	77.0
DeCAL 8x	32.2	16.2	29.8	60.3	74.4
CCProxy 2x	33.6	17.2	30.5	59.2	73.6
CCProxy 4x	33.1	16.6	29.9	56.9	71.0
CCProxy 8x	31.9	15.5	29.0	55.1	69.0
AttnPool 2x	34.1	17.5	31.1	60.2	74.4

Table 4: ROUGE metrics on MediaSum and SQuAD metrics on HotpotQA for different levels of compression for both DeCAL and CCProxy against no compression and AttnPool 2x.

Due to resource limitations, we chose one summarization and one question-answering task for our comparisons. We selected MediaSum and HotpotQA, since they showed greatest sensitivity to compression ratio.

From table 4, we see that CCProxy is consistently outperformed by DeCAL at the same compression ratio; in fact, for MediaSum, CCProxy largely performs similarly to DeCAL at twice its compression ratio. For HotpotQA, this task proves more difficult with CCProxy 2x coming in worse than DeCAL 8x. The spread in metrics between 2x and 8x is actually slightly smaller for CCProxy than for DeCAL. We can attribute the difference in performance directly to the CCProxy elements outlined above.

AttnPool 2x comes very close to DeCAL 4x on MediaSum and close to DeCAL 8x on HotpotQA. This is a strong indicator that the extra compute cost from adding the latent sequence in DeCAL does translate into superior compressed representations.

4.3 ENCODER-ONLY, RETRIEVAL EXPERIMENTS

ColBERT’s (Khattab & Zaharia, 2020) late interaction approach offers superior retrieval to single-vector by virtue of allowing each query token to interact with each passage token, but suffers from dramatically greater compute cost (a dot product for each interaction). Pruning techniques have been proposed (Santhanam et al., 2022; Qian et al., 2022) to address the compute cost, but pruning loses information and does not save storage cost. In contrast, DeCAL does not drop any tokens, so information should be lost more gracefully as compression is increased, and by reducing the representation up-front, storage costs are reduced. We do not have a setup for reproducing either pruning technique for comparison (left for future work), but we include these results to illustrate the versatility of DeCAL and compare against the uncompressed baseline.

For these experiments, we took an already pretrained (as described under Setup) DeCAL model at some compression C, discarded the decoder and fine-tuned the encoder in a dual encoder configuration using T5X Retrieval². We utilized Chamfer similarity loss as in ColBERT (Khattab & Zaharia, 2020).

²https://github.com/google-research/t5x_retrieval

4.3.1 DATASETS

We used MS MARCO (Nguyen et al., 2016) as the retrieval fine-tuning dataset (for 12M examples). For evaluation, we used a subset of the BEIR (Thakur et al., 2021) retrieval benchmark: FiQA-2018, TREC-COVID, Touché-2020, NFCorpus, and SciFact, and we report NDCG@10. These (smaller) datasets were selected simply due to resource constraints we had for these experiments.

4.3.2 RESULTS

	FiQA	SciFact	NFCorpus	Touché	COVID
	NDCG@10				
Baseline	0.346	0.709	0.348	0.249	0.612
DeCAL 2x	0.359	0.703	0.349	0.299	0.685
DeCAL 4x	0.320	0.681	0.338	0.274	0.709
DeCAL 8x	0.301	0.658	0.337	0.232	0.671

Table 5: Retrieval results for different levels of DeCAL compression and without DeCAL for 5 BEIR datasets.

In Table 5, we see that DeCAL 2x is as good as or better than the baseline, with a 4x computation savings. DeCAL 4x performance drops off only modestly for 16x computation savings. DeCAL 8x drops off further, but at 64x in computation savings and 8x reduction in storage versus the baseline.

Interestingly, we see a wide dispersion of response to compression in the NDCG@10 metric. FiQA is hurt relatively most, but COVID appears to consistently benefit even to 8x compression. We also see that 2x compression is frequently better than no compression; we speculate that compression may be fusing subwords such that the combined information actually improves retrieval. Doing this more intelligently may reap further benefits.

Intuitively, compressing the query too much can hurt multi-vector retrieval since we lose conceptual granularity. We might obtain better results at high compression using an asymmetric dual encoder with low compression on the query, but that is left for future work.

4.4 ABLATIONS

In this section we focus on HotpotQA, since it showed the greatest sensitivity to compression. We also use 2x compression since that represents the upper bound of performance under compression.

	HotpotQA	
	EM	F1
Baseline	65.1	79.2
DeCAL 2x	65.2	79.5
DeCAL 2x (autoencode)	63.9	78.3
DeCAL 2x (fine-tune-only)	62.6	78.0
DeCAL 2x (no pooling)	61.8	75.8
AttnPool 2x	60.2	74.4

Table 6: Ablations: SQuAD metrics for HotpotQA comparing (1) the vanilla T5 baseline, (2) DeCAL 2x, (3) denoising replaced by autoencoding for DeCAL 2x, (4) applying DeCAL 2x during fine-tune (but not pretrain), (5) omitting token pooling for the initial latent in DeCAL 2x, and (6) AttnPool 2x.

There are three main components of DeCAL that we can ablate: (1) language modeling pretraining with a compressive encoder, (2) the latent-based DeCAL compressive structure itself, and (3) the token pooling when forming the initial input latent sequence.

For (1), we can omit the denoising task entirely and first introduce compression during fine-tuning (on top of the vanilla baseline). Alternatively, we can train compression with autoencoding (mixed 1:1 with text continuation to avoid overfitting on reconstruction) in place of denoising, as we did for CCProxy. In Table 6, we see that both of these yield a result that is notably worse than when pretrained with denoising and so can no longer hold up against the baseline.

For (2), in place of the DeCAL encoder structure, we can instead apply the commonly-used form of attention pooling by applying it on the final encoder output. This is what we already included in Comparisons as AttnPool 2x. Included here in context with the other ablations, we can see that this is actually the most damaging of them. We conclude that building up the compressed representation over many layers rather than in one is critical for optimal compression.

For (3), When we use only the learnable vector v and skip the initial token pooling, we see that it hurts performance significantly. This indicates that including the representations of the input tokens plays a useful role in guiding the information transfer from the input tokens to the smaller latent sequence.

These ablations give clear support to the value of the multilayer latent-based compressive structure, denoising pretraining, and initial token pooling as important facets of the DeCAL design.

5 CONCLUSION

In this paper, we introduced DeCAL, a novel method for tokenwise compression. We demonstrated that the compressed representations can be very usable, matching baseline at 2x compression on all tasks tested. At a significant 8x compression, average ROUGE-L for DeCAL is only 4.1% lower than baseline for summarization tasks, and average F1 is only 4.3% lower for question-answering tasks. Similarly, average NDCG@10 is only 2.9% lower than baseline on multi-vector retrieval tasks at 8x compression, and demonstrates the versatility of DeCAL over a wide range of downstream tasks.

We extracted the key common elements of recent LLM context compressors to construct a proxy for comparison. DeCAL outperformed on both summarization and question-answering tasks, highlighting the advantages of the combination of denoising pretraining, initial token pooling for the latent sequence, and bidirectional attention. We also showed a strong benefit from utilizing multiple layers of computation on the compressed latent sequence alongside the full input sequence in contrast to simply attention pooling the encoder output to achieve compression.

There are many avenues for further exploration. We found the default T5 (Raffel et al., 2020) span corruption pretrain tasks worked well, but a task tailored for compression may work even better. We have not evaluated the balance of encoder and decoder size, compressing sequences longer than 4k tokens, or more sophisticated x -token to I -token mappings for initial pooling to facilitate variable compression. DeCAL may be applicable to character/byte-level models and multimodal sources. Work remains to explore techniques to push DeCAL compression to 16x and beyond. We have yet to try Retrieval-Augmented Generation, a common target for context compression, where we can take advantage of pre-computed compressed passages to fill large contexts. Lastly, we can seek ways to maintain compression quality while reducing compute.

ACKNOWLEDGMENTS

We would like to thank the Gemini Embeddings team for assistance with multi-vector retrieval. We also thank Joshua Ainslie for past work on sparse attention, which inspired experiments that ultimately led to the development of DeCAL.

REFERENCES

- Aydar Bulatov, Yuri Kuratov, Yermek Kapushev, and Mikhail S. Burtsev. Scaling transformer to 1m tokens and beyond with rmt, 2024. URL <https://arxiv.org/abs/2304.11062>.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=kplU6wBPXq>.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 01 2022. ISSN 2307-387X. doi: 10.1162/tacl_a_00448. URL https://doi.org/10.1162/tacl_a_00448.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097. URL <https://aclanthology.org/N18-2097/>.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. Funnel-transformer: filtering out sequential redundancy for efficient language processing. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS ’20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423/>.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=uREj4ZuGJE>.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh M. Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. Power-bert: accelerating bert inference via progressive word-vector elimination. In *Proceedings of the 37th International Conference on Machine Learning*, ICML’20. JMLR.org, 2020.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. LongT5: Efficient text-to-text transformer for long sequences. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 724–736, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.55. URL <https://aclanthology.org/2022.findings-naacl.55/>.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/jaegle21a.html>.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL <https://aclanthology.org/P17-1147/>.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’20, pp. 39–48, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075. URL <https://doi.org/10.1145/3397271.3401075>.

-
- Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226, 2018. URL <http://arxiv.org/abs/1808.06226>.
- Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *CoRR*, abs/2101.00190, 2021. URL <https://arxiv.org/abs/2101.00190>.
- Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gülcöhre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Stefan Riezler and Yoav Goldberg (eds.), *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL <https://aclanthology.org/K16-1028/>.
- Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Lukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. *CoRR*, abs/2110.13711, 2021. URL <https://arxiv.org/abs/2110.13711>.
- Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. November 2016. URL <https://www.microsoft.com/en-us/research/publication/ms-marco-human-generated-machine-reading-comprehension-dataset/>.
- Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. Byte latent transformer: Patches scale better than tokens, 2024. URL <https://arxiv.org/abs/2412.09871>.
- Yujie Qian, Jinhyuk Lee, Sai Meher Karthik Duddu, Zhuyun Dai, Siddhartha Brahma, Iftekhar Naim, Tao Lei, and Vincent Y. Zhao. Multi-vector retrieval as sparse alignment. *CoRR*, abs/2211.01267, 2022. doi: 10.48550/ARXIV.2211.01267. URL <https://doi.org/10.48550/arXiv.2211.01267>.
- Guanghui Qin and Benjamin Van Durme. Nugget: neural agglomerative embeddings of text. In *Proceedings of the 40th International Conference on Machine Learning*, ICML’23. JMLR.org, 2023.
- Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. *CoRR*, abs/1911.05507, 2019. URL <http://arxiv.org/abs/1911.05507>.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.
- David Rau, Shuai Wang, Hervé Déjean, Stéphane Clinchant, and Jaap Kamps. Context embeddings for efficient answer generation in retrieval-augmented generation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, WSDM ’25, pp. 493–502, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713293. doi: 10.1145/3701551.3703527. URL <https://doi.org/10.1145/3701551.3703527>.
- Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. Plaid: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM ’22, pp. 1747–1756, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392365. doi: 10.1145/3511808.3557325. URL <https://doi.org/10.1145/3511808.3557325>.
- Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhijun Tu, Kai Han, Hailin Hu, and Dacheng Tao. A Survey on Transformer Compression. *arXiv e-prints*, art. arXiv:2402.05964, February 2024. doi: 10.48550/arXiv.2402.05964.
- Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6), December 2022. ISSN 0360-0300. doi: 10.1145/3530811. URL <https://doi.org/10.1145/3530811>.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=wCu6T5xFjeJ>.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fdb053c1c4a845aa-Paper.pdf.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October–November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL <https://aclanthology.org/D18-1259/>.

Chenguang Zhu, Yang Liu, Jie Mei, and Michael Zeng. MediaSum: A large-scale media interview dataset for dialogue summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5927–5934, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.474. URL <https://aclanthology.org/2021.naacl-main.474/>.